Proyecto de Programación en Ensamblador
Estructura de Computadores
Grado en Ingeniería Informática

Ejemplos de casos de prueba

Departamento de Arquitectura y Tecnología de Sistemas Informáticos

2020-2021 (primer semestre)

Este documento complementa la sección "Ejemplos" del enunciado del proyecto de programación en ensamblador (pág. 22). Contiene varios ejemplos de casos de prueba para cada una de las subrutinas que componen el proyecto.

Estos ejemplos no constituyen un juego de ensayo completo. No sirven por sí solos para comprobar el correcto funcionamiento de las subrutinas ni para detectar de forma completa en qué situaciones presentan comportamientos erróneos. Sin embargo, se pueden seguir como guía para la elaboración de juegos de ensayo más completos.

En cada ejemplo se presenta el contenido de la memoria principal antes y después de la ejecución de una subrutina. Este contenido se muestra tal y como lo hace el simulador del 88110 usando el comando "V". En concreto, se muestra lo siguiente:

- Los datos de prueba suministrados como parámetros a las subrutinas.
- El contenido de la pila al comienzo de su ejecución. En el caso de FiltRec también se muestra su contenido final incluyendo los parámetros de salida NCambios y MaxFiltrados.
- Los resultados que producen.

En algunos casos también se describen los datos de prueba en lenguaje ensamblador. En otros en su descripción se indica "Los registros parten de valores distintos de 0"; en estos casos, antes de la llamada a la subrutina, se inicializan los registros con valores arbitrarios.

Por último, en la página 20, se incluye el programa de prueba de un caso concreto para la subrutina Comp.

En este procesador el direccionamiento se hace a nivel de byte y se utiliza el formato *little-endian*. En consecuencia, cada una de las palabras representadas a continuación de la especificación de la dirección debe interpretarse como formada por 4 bytes con el orden que se muestra en el ejemplo siguiente:

```
Direcciones de memoria, tal como las muestra el simulador:
60000       04050607     05010000

Direcciones de memoria, tal como se deben interpretar:
60000       04
60001       05
60002       06
60003       07

60004       05
60005       01
60006       00
60007       00

Valor de las palabras almacenadas en las posiciones 60000 y
60004, tal como la interpreta el procesador:

60000       0x07060504 = 117.835.012
60004       0x00000105 = 261
```

Raíz cuadrada

Caso 1. Llamada a `Sqrt1d`

Llama a 'Sqrt1d' pasándole el parámetro 99

```
r30=86012 (0x14FFC)
Direcciones de memoria:

86000                                                    63000000
```

Resultado:

```
r30=86012 (0x14FFC) r29 = 99 (0x63)
```

Caso 2. Llamada a `Sqrt1d`

Llama a 'Sqrt1d' pasándole un parámetro que es un cuadrado perfecto de un entero mayor que 1.000

```
r30=86012 (0x14FFC)
Direcciones de memoria:

86000                                                    69690F00
```

Resultado:

```
r30=86012 (0x14FFC) r29 = 10050 (0x2742)
```

Número de filtrados

Caso 3. Llamada a `nFiltrados`

Llama a 'nFiltrados' pasándole un parámetro nulo para iniciar la variable nF, que tiene un valor no nulo.

```
r30=86012 (0x14ffc)
   Direcciones de memoria:
   00000      0E000000

   86000                                                  00000000
```

Resultado:

```
r30=86012 (0x14ffc) r29=0 (0x00)
   Direcciones de memoria:
   00000      0000000
```

Caso 4. Llamada a `nFiltrados`

Llama a 'nFiltrados' pasándole un parámetro negativo para incrementar la variable nF, que tiene un valor positivo.

```
r30=86012 (0x14ffc)
   Direcciones de memoria:
   00000      0E000000

   86000                                                  FFFFFFFF
```

Resultado:

```
r30=86012 (0x14ffc) r29=15 (0xF)
   Direcciones de memoria:
   00000      0F000000
```

Compara dos imágenes

Caso 5. Llamada a `Comp`

Llama a 'Comp' pasándole dos imagenes de 4x8 elementos que difieren en uno solo de ellos.

```
        org 0x14000
IMAGEN1:
        data    4, 8
        data    0x00000000, 0x00000000
        data    0x00000000, 0x00002100
        data    0x00000000, 0x00000000
        data    0x00000000, 0x00000000
IMAGEN2:
        data    4, 8
        data    0x00000000, 0x00000000
        data    0x00000000, 0x00000000
        data    0x00000000, 0x00000000
        data    0x00000000, 0x00000000
```

```
r30=86008 (0x14ff8)
   Direcciones de memoria:
   86000                                00400100     28400100

   81920      04000000     08000000     00000000     00000000
   81936      00000000     00210000     00000000     00000000
   81952      00000000     00000000

   81952                                04000000     08000000
   81968      00000000     00000000     00000000     00000000
   81984      00000000     00000000     00000000     00000000
```

Resultado:

```
r30=86008 (0x14FF8) r29=330 (0x014a)
```

Caso 6. Llamada a `Comp`

Llama a 'Comp' pasándole dos imágenes de 4x8 elementos en las que difieren todos sus elementos en una o en dos unidades.

```
r30=86008 (0x14ff8)
   Direcciones de memoria:
   86000                                 00400100     28400100

   81920      04000000     08000000      55FF55FF     55FF55FF
   81936      FF55FF55     FF55FF55      55FF55FF     55FF55FF
   81952      FF55FF55     FF55FF55

   81952                                 04000000     08000000
   81968      54FE54FE     54FE54FE      FD57FD57     FD57FD57
   81984      54FE54FD     54FE54FE      FD53FD53     FD53FD53
```

Resultado:

```
r30=86008 (0x14ff8) r29=91 (0x5B)
```

Caso 7. Llamada a `Comp`

Llama a 'Comp' pasándole dos imágenes diferentes de 4x8 elementos. Se observan diferencias, en algunos casos negativas y en otros positivas, en varios píxeles de las imágenes, tales que en una de ellas se usa un valor por encima de 128 y en la otra por debajo de 128.

```
r30=86008 (0x14ff8)
   Direcciones de memoria:
   86000                                 00400100     28400100

   81920      04000000     08000000      000000D1     00000000
   81936      00000000     1D7A0000      00000001     00000000
   81952      00000000     FF000000

   81952                                 04000000     08000000
   81968      00000002     00000000      00000000     82010000
   81984      000000FF     00000000      00000000     10000000
```

Resultado:

```
r30=86008 (0x14ff8) r29=4351 (0x10ff)
```

Extracción de submatriz

Caso 8. Llamada a `SubMatriz`

Llama a 'SubMatriz' pasándole una imagen de 3x3 elementos de la que se ha de extraer la subimagen correspondiente al elemento central.

```
        org 0x14000
IMAGEN:
        data    3, 3
        data    0x40302010, 0x80706050, 0x90

        org 0x14040
SUBIMAGEN:
        data    0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF
```

```
r30=86000 (0x14ff0)
   Direcciones de memoria:
   86000      00400100     40400100     01000000     01000000

   81920      03000000     03000000     10203040     50607080
   81936      90000000

   81984      FFFFFFFF     FFFFFFFF     FFFFFFFF
```

Resultado:

```
r30=86000 (0x14ff0)
   Direcciones de memoria:
   81984      10203040     50607080     90FFFFFF
```

Caso 9. Llamada a `SubMatriz`

Llama a 'SubMatriz' pasándole una imagen de 5x8 elementos de la que se ha de extraer la subimagen correspondiente a uno de los píxeles del interior de la imagen.
Los registros parten de valores distintos de 0.

```
r30=86000 (0x14ff0)
   Direcciones de memoria:
   86000      00400100     40400100     03000000     06000000

   81920      05000000     08000000     01020304     05060708
   81936      090A0B0C     0D0E0F10     11121314     15161718
   81952      191A1B1C     1D1E1F20     21222324     25262728

   81984      FFFFFFFF     FFFFFFFF     FFFFFFFF
```

Resultado:

```
r30=86000 (0x14ff0)
   Direcciones de memoria:
   81984      1617181E     1F202627     28FFFFFF
```

Valor del píxel filtrado

Caso 10. Llamada a `ValorPixel`

Llama a 'ValorPixel' pasándole una subimagen nula excepto en su elemento central y un filtro identidad.

```
        org     0x14000
SUBIMAGEN:
        data    0x00000000, 0x00000055, 0x00

        org     0x14010
FILTRO: data    0, 1, 0, 1, 0, 1
        data    0, 1, 1, 1, 0, 1
        data    0, 1, 0, 1, 0, 1

r30=86008 (0x14FF8)
   Direcciones de memoria:
   86000                                00400100     10400100

   81920      00000000     55000000     00000000

   81936      00000000     01000000     00000000     01000000
   81952      00000000     01000000     00000000     01000000
   81968      01000000     01000000     00000000     01000000
   81984      00000000     01000000     00000000     01000000
   82000      00000000     01000000
```

Resultado:

```
r30=86008 (0x14FF8) r29=85 (0x55)
```

Caso 11. Llamada a `ValorPixel`

Llama a 'ValorPixel' pasándole una subimagen no nula y un filtro que dobla y cambia el signo del elemento al que se aplica.

```
r30=86008 (0x14FF8)
   Direcciones de memoria:
   86000                                 00400100     10400100

   81920      00000000     55000000      00000000

   81936      00000000     01000000      00000000     01000000
   81952      00000000     01000000      00000000     01000000
   81968      02000000     FFFFFFFF      00000000     01000000
   81984      00000000     01000000      00000000     01000000
   82000      00000000     01000000
```

Resultado:

```
r30=86008 (0x14FF8) r29=-170 (0xFFFFFF56)
```

Caso 12. Llamada a `ValorPixel`

Llama a 'ValorPixel' pasándole una subimagen no nula y un filtro que devuelve el valor negativo del doble de la suma de los ocho elementos que lo rodean.
Los registros parten de valores distintos de 0.

```
r30=86008 (0x14FF8)
   Direcciones de memoria:
   86000                                 00400100     10400100

   81920      10111213     14151617      18000000

   81936      02000000     FFFFFFFF      FEFFFFFF     01000000
   81952      02000000     FFFFFFFF      FEFFFFFF     01000000
   81968      00000000     01000000      FEFFFFFF     01000000
   81984      02000000     FFFFFFFF      FEFFFFFF     01000000
   82000      02000000     FFFFFFFF
```

Resultado:

```
r30=86008 (0x14FF8) r29=-320 (0xFFFFFEC0)
```

Filtro de un píxel

Caso 13. Llamada a `FilPixel`

Llama a 'FilPixel' pasándole una imagen de 5x5 elementos y un filtro identidad que se aplica a un píxel del interior de la imagen.

```
        org     0x14000
IMAGEN:
        data    5, 5
        data    0x44332211, 0x03020155
        data    0x22210504, 0x31252423
        data    0x35343332, 0x44434241
        data    0x00000045

FILTRO: data    0, 1,  0,  1, 0, 1
        data    0, 1, -5, -5, 0, 1
        data    0, 1,  0,  1, 0, 1


r30=86000 (0x14FF0)
   Direcciones de memoria:
   86000      00400100     02000000     03000000     24400100

   81920      05000000     05000000     11223344     55010203
   81936      04052122     23242531     32333435     41424344
   81952      45000000

   81952                   00000000     01000000     00000000
   81968      01000000     00000000     01000000     00000000
   81984      01000000     FBFFFFFF     FBFFFFFF     00000000
   82000      01000000     00000000     01000000     00000000
   82016      01000000     00000000     01000000
```

Resultado:

```
r30=86000 (0x14FF0) r29=36 (0x24)
```

Caso 14. Llamada a `FilPixel`

Llama a 'FilPixel' pasándole una imagen de 4x8 elementos y un filtro que devuelve la media de los ocho elementos que rodean al píxel seleccionado.

```
r30=86000 (0x14FF0)
   Direcciones de memoria:
   86000      00400100     02000000     02000000     30400100

   81920      04000000     08000000     44444444     44444444
   81936      44343433     44444444     44448844     44444444
   81952      44444444     44444444

   81968      01000000     08000000     01000000     08000000
   81984      01000000     08000000     01000000     08000000
   82000      00000000     08000000     01000000     08000000
   82016      01000000     08000000     01000000     08000000
   82032      01000000     08000000
```

Resultado:

```
r30=86000 (0x14FF0) r29=61 (0x3D)
```

Caso 15. Llamada a `FilPixel`

Llama a 'FilPixel' pasándole una imagen de 4x8 elementos y un filtro que multiplica por -8 el valor del píxel seleccionado y le suma el valor de los ocho píxeles que lo rodean. El resultado se ajusta al valor mínimo (0).
Los registros parten de valores distintos de 0.

```
r30=86000 (0x14FF0)
   Direcciones de memoria:
   86000      00400100     02000000     02000000     28400100

   81920      04000000     08000000     43424140     47464544
   81936      4B4A4948     4F4E4D4C     43429940     47464544
   81952      4B4A4948     4F4E4D4C

   81952                                01000000     01000000
   81968      01000000     01000000     01000000     01000000
   81984      01000000     01000000     F8FFFFFF     01000000
   82000      01000000     01000000     01000000     01000000
   82016      01000000     01000000     01000000     01000000
```

Resultado:

```
r30=86000 (0x14FF0) r29=0 (0x00000000)
```

Filtro de imagen

Caso 16. Llamada a `Filtro`

Llama a 'Filtro' pasándole una imagen no nula de 4x8 elementos y un filtro que multiplica por 4 cada elemento y le resta tres veces el valor del situado en la misma columna de la fila anterior. Algunos elementos alcanzan el valor máximo y otros el mínimo.

```
        org 0x14000
IMAGEN:
        data    4, 8
        data    0x04030201, 0x07060504
        data    0x14134211, 0x17168514
        data    0x24232221, 0x27262574
        data    0x34333231, 0x37363534
FILTRADA:
        res     40
FILTRO: data    0, 1, -3, 1, 0, 1
        data    0, 1,  4, 1, 0, 1
        data    0, 1,  0, 1, 0, 1
```

```
r30=86004 (0x14FF4)
   Direcciones de memoria:
   86000                   00400100     28400100     50400100

   81920      04000000     08000000     01020304     04050607
   81936      11421314     14851617     21222324     74252627
   81952      31323334     34353637

   81952                                00000000     00000000
   81968      00000000     00000000     00000000     00000000
   81984      00000000     00000000     00000000     00000000

   82000      00000000     01000000     FDFFFFFF     01000000
   82016      00000000     01000000     00000000     01000000
   82032      04000000     01000000     00000000     01000000
   82048      00000000     01000000     00000000     01000000
   82064      00000000     01000000
```

Resultado:

```
r30=86004 (0x14FF4)
   Direcciones de memoria:
   81952                                04000000     08000000
   81968      01020304     04050607     11FF4344     44FF4617
   81984      21005354     FF005627     31323334     34353637
```

Caso 17. Llamada a `Filtro`

Llama a 'Filtro' pasándole una imagen de 4x6 elementos y un filtro que sustituye cada píxel por la media de los que están situados en los cuatro vértices de la submatriz que lo rodea. El filtro utiliza coeficientes negativos.

```
r30=86004 (0x14FF4)
   Direcciones de memoria:
   86000                 00400100    20400100    40400100

   81920     04000000    06000000    01020304    05060002
   81936     04010305    0306090C    0F120408    10204080

   81952     A5A5A5A5    A5A5A5A5    A5A5A5A5    A5A5A5A5
   81968     A5A5A5A5    A5A5A5A5    A5A5A5A5    A5A5A5A5

   81984     FEFFFFFF    F8FFFFFF    00000000    F8FFFFFF
   82000     FEFFFFFF    F8FFFFFF    00000000    F8FFFFFF
   82016     00000000    F8FFFFFF    00000000    F8FFFFFF
   82032     FEFFFFFF    F8FFFFFF    00000000    F8FFFFFF
   82048     FEFFFFFF    F8FFFFFF
```

Resultado:

```
r30=86004 (0x14FF4)
   Direcciones de memoria:
   81952     04000000    06000000    01020304    05060004
   81968     06080A05    03060A15    29120408    10204080
```

Caso 18. Llamada a `Filtro` Llama a 'Filtro' pasándole una imagen no nula de 4x6 elementos y un filtro identidad, que devuelve la misma imagen recibida.

```
r30=86004 (0x14FF4)
   Direcciones de memoria:
   86000                  00400100     20400100     40400100

   81920      04000000    06000000     78563412     FCFDFEFF
   81936      79573513    EBECEDEE     89674523     DCDDDEDF

   81952      00000000    00000000     04030201     02010605
   81968      06050403    04030201     02010605     06050403

   81984      00000000    01000000     00000000     01000000
   82000      00000000    01000000     00000000     01000000
   82016      01000000    01000000     00000000     01000000
   82032      00000000    01000000     00000000     01000000
   82048      00000000    01000000
```

Resultado:

```
r30=86004 (0x14FF4)
   Direcciones de memoria:
   81952      04000000    06000000     78563412     FCFDFEFF
   81968      79573513    EBECEDEE     89674523     DCDDDEDF
```

Filtro recursivo de una imagen

Caso 19. Llamada a `FiltRec`

Llama a 'FiltRec' sobre una imagen de 4x4 elementos, con un filtro que devuelve para cada píxel la media de los que lo rodean. El parámetro NCambios tiene valor 400 y MaxFiltrados 4.

```
        org 0x14000
IMAGEN:
        data    4, 4
        data    0x04030201, 0x0D0E0F10, 0x05040302, 0x23222120
FILTRO: data    1, 8, 1, 8, 1, 8
        data    1, 8, 0, 8, 1, 8
        data    1, 8, 1, 8, 1, 8
FILTRADA:   res     24
        data    0x55555555, 0x55555555
r30=85996 (0x14FEC)
   Direcciones de memoria:
   00000      00000000

   85984                                           00400100
   86000      60400100     18400100     90010000     04000000

   81920      04000000     04000000     01020304     100F0E0D
   81936      02030405     20212223

   81936                                01000000     08000000
   81952      01000000     08000000     01000000     08000000
   81968      01000000     08000000     00000000     08000000
   81984      01000000     08000000     01000000     08000000
   82000      01000000     08000000     01000000     08000000

   82016      00000000     00000000     00000000     00000000
   82032      00000000     00000000

   82032                                55555555     55555555
```

Resultado:

```
r30=86996 (0x14FEC)  r29=0 (0x00)
   Direcciones de memoria:
   00000      01000000

   85984                                           00400100
   86000      60400100     18400100     F7000000     01000000

   82016      04000000     04000000     01020304     1005060D
   82032      02121305     20212223

   82032                                55555555     55555555
```

Caso 20. Llamada a `FiltRec`

Llama a 'FiltRec' sobre una imagen de 5x5 elementos, con un filtro que devuelve para cada píxel la mitad de su valor. El parámetro NCambios tiene valor 20 y MaxFiltrados 5. Los registros parten de valores distintos de cero.

```
r30=85996 (0x14FEC)
   Direcciones de memoria:

   00000      00000000

   85984                                              00400100
   86000      00410100     24400100     14000000     05000000

   81920      05000000     05000000     0A000A00     0A000000
   81936      00000A00     0A000A00     00000000     0A000A00
   81952      0A000000

   81952                   00000000     01000000     00000000
   81968      01000000     00000000     01000000     00000000
   81984      01000000     01000000     02000000     00000000
   82000      01000000     00000000     01000000     00000000
   82016      01000000     00000000     01000000

   82176      00000000     00000000     00000000     00000000
   82192      00000000     00000000     00000000     00000000
   82208      00000000

   82208                    A5A5A5A5
```

Resultado:

```
r30=85996 (0x14FEC)  r29=0 (0x00)
   Direcciones de memoria:

   00000      03000000

   85984                                              00400100
   86000      00410100     24400100     0A000000     03000000

   82176      05000000     05000000     0A000A00     0A000000
   82192      00000A00     01000A00     00000000     0A000A00
   82208      0A000000

   82208                    A5A5A5A5
```

Caso 21. Llamada a `FiltRec`

Llama a 'FiltRec' sobre una imagen de 4x8 elementos, con un filtro que sustituye cada píxel por la media de los que están situados en los cuatro vértices de la submatriz que lo rodea. El parámetro NCambios tiene valor 0 y MaxFiltrados 2.
Los registros parten de valores distintos de cero.

```
r30=85996 (0x14FEC)
   Direcciones de memoria:

   00000      00000000

   85984                                          00400100
   86000      7C400100     30400100     00000000     02000000

   81920      04000000     08000000     FF0000FF     FF0000FF
   81936      FF0000FF     FF0000FF     FF0000FF     FF0000FF
   81952      FF0000FF     FF0000FF

   81968      01000000     04000000     00000000     04000000
   81984      01000000     04000000     00000000     04000000
   82000      00000000     04000000     00000000     04000000
   82016      01000000     04000000     00000000     04000000
   82032      01000000     04000000

   82032                                          00000000
   82048      00000000     00000000     00000000     00000000
   82064      00000000     00000000     00000000     00000000
   82080      00000000

   82084                   AAAAAAAA     AAAAAAAA
```

Resultado:

```
r30=85996 (0x14FEC)  r29=-1 (0xFFFFFFFF)
   Direcciones de memoria:

   00000      02000000

   85984                                          00400100
   86000      7C400100     30400100     80020000     02000000

   82032                                          04000000
   82048      08000000     FF0000FF     FF0000FF     FF9F7F7F
   82064      7F7F9FFF     FF9F7F7F     7F7F9FFF     FF0000FF
   82080      FF0000FF

   82084                   AAAAAAAA     AAAAAAAA
```

```
;
;           Ejemplo de programa de prueba de la subrutina Comp
;

LEA:    MACRO   (ra, eti)
        or ra, r0, low(eti)
        or.u ra, ra, high(eti)
        ENDMACRO

PUSH:   MACRO(ra)
        subu r30, r30, 4
        st ra, r30, 0
        ENDMACRO

POP:    MACRO(ra)
        ld ra, r30, 0
        addu r30, r30, 4
        ENDMACRO

; Definición de la pila
        org     0xF000
PILA:   data    0

; Definición de las imágenes de prueba (2x2)
        org     0x1000
IMG1:   data    0x02, 0x02, 0x07050301

        org     0x2000
IMG2:   data    0x02, 0x02, 0x04030201

; Programa principal
        org     100
PPAL:   LEA (r30, PILA)     ; Inicialización del puntero de pila
        LEA (r1, IMG1)      ; r1: dirección de comienzo de IMG1
        LEA (r2, IMG2)      ; r2: dirección de comienzo de IMG2

        PUSH (r2)           ; Paso de parámetro Imagen2
        PUSH (r1)           ; Paso de parámetro Imagen1

        bsr Comp            ; Llamada a la subrutina Comp
                            ; r29 debe valer 37 (0x000000025)

        POP (r1)
        POP (r2)

        stop


Comp:   PUSH (r1)
;        ....
```